Экзаменационный проект
по дисциплине
Проектирование баз данных
студента гр. **M34391**
Кононова Артемия Владимировича
по теме
youtube

Описание проекта

База данных сайта youtube.com.

Построение отношений

В результате предварительного проектирования были выделены следующие отношения:

- <<Users>> — пользователи. Имеют возможно повторяющиеся UserName.
- <<Videos>> — загруженные пользователями видео.
- <<Comments>> — оставленные пользователями комментарии к видео. Любой пользователь может оставить комментарий к любому видео.
- <<Video Likes>> — выставленные пользователями оценки для видео.
- <<Comment Likes>> — выставленные пользователями оценки для комментариев.
- <<Playlists created by User>> — созданные пользователями плейлисты.
- <<Added Videos>> — видео, добавленные в плейлисты.

Отношение <<Users>>

Атрибуты:

- <<UserId>> — идентификатор пользователя
- <<Username>> — отображаемое имя

Функциональные зависимости:

- UserId -> Username

Ключи:

- UserId

Отношение <<Comments>>

Атрибуты:

- <<CommentId>> — идентификатор комментария
- <<UserId>> — идентификатор пользователя, который оставил комментарий
- <<VideoId>> — идентификатор видео, к которому оставлен комментарий
- <<CommentText>> — текст комментария
- <<CreatedAt>> — дата комментария

Функциональные зависимости:

- CommentId -> UserId
- CommentId -> Videoid
- CommentId -> CommentText
- CommentId -> CreatedAt

Ключи:

- CommentId

Отношение <<Videos>>

Атрибуты:

- <<VideoId>> — идентификатор видео
- <<UserId>> — идентификатор пользователя, который загрузил видео
- <<VideoName>> — название видео
- <<VideoDescription>> — описание видео
- <<UploadedAt>> — дата загрузки видео
- <<VideoLength>> — длительность видео

Функциональные зависимости:

- VideoId -> UserId
- VideoId -> VideoName
- VideoId -> VideoDescription
- VideoId -> UploadedAt
- VideoId -> VideoLength

Ключи:

- VideoId

Отношение <<Comment Likes>>

Атрибуты:

- <<CommentId>> — идентификатор комментария, которому выставлена оценка
- <<UserId>> — идентификатор пользователя, который выставил оценку
- <<IsLike>> — значение оценки - нравится/не нравится

Функциональные зависимости:

- UserId, CommentId -> IsLike

Ключи:

- (UserId, CommentId)

Отношение <<Video Likes>>

Атрибуты:

- <<VideoId>> — идентификатор видео, которому выставлена оценка
- <<UserId>> — идентификатор пользователя, который выставил оценку
- <<IsLike>> — значение оценки - нравится/не нравится

Функциональные зависимости:

- UserId, VideoId -> IsLike

Ключи:

- (UserId, VideoId)

Нормализация этого отношения:

все атрибуты атомарны и нету повторяющихся групп - отношение в 1-НФ
1ФЗ, и в ней IsLike зависит от ключа целиком - отношение в 2-НФ
IsLike зависит от ключа напрямую - отношение в 3-НФ
IsLike зависит от ключа - отношение в НФБК

4-НФ - для любой нетривиальной МЗ X->Y|Z, X - надключ. В отношении лишь 3 атрибута, честно перебрав все варианты X->Y|Z в МЗ можно убедиться, что все МЗ - тривиальны, значит отношение в 4НФ.
Ниже приведен перебор таких вариантов с пояснениями:
VideoId -> UserId | IsLike - не мз, тк при разных IsLike множество UserId разнится.
VideoId -> IsLike | UserId - не мз, тк при разных Userid множество IsLike разнится.
IsLike -> VideoId | UserId - не мз, тк при разных UserId множество VideoId разное
IsLike -> UserId| VideoId - не мз, тк при разных VideoId множество UserId разное
UserId -> IsLike| VideoId - не мз, тк для разных VideoId множество IsLike разное
UserId -> VideoId| IsLike - не мз, тк для разных IsLike множество VideoId разное

5-НФ - для каждой нетривиальной ЗС каждый элемент декомпозиции - надключ. Честно переберем все возможные декомпозиции исходного отношения и проверим, что они могут быть ЗС. Ниже приведены все варианты и пояснения к ним.
Ниже приведен перебор таких вариантов с пояснениями:
R(VideoId, UserId, IsLike) =
= R(VideoId) + R(UserId) + R(IsLike) - не ЗС, тк в R(IsLike) содержатся и true и false - при объединении обратно мы получим и лайк и дизлайк одновременно.
= R(VideoId, UserId) + R(IsLike) - не ЗС, то же самое что в прошлом варианте
= R(VideoId, IsLike) + R(UserId) - не ЗС, тк при объединении мы получим, что каждому UserId сопоставлено (VideoId, IsLike) - пользователь лайкнул каждое видео - это не эквивалентное отношение
= R(UserId, IsLike) + R(VideoId) - не ЗС, тк при объединении оценка пользователя будет поставлена каждому видео. Снова не эквивалентное отношение
= R(VideoId, UserId) + R(VideoId, IsLike) - не ЗС, тк если у видео есть лайк и дизлайк одновременно - при объединении эта оценка распространится на каждого пользователя, оставившего оценку этому видео - не эквивалентное отношение
= R(UserId, VideoId) + R(UserId, IsLike) - не ЗС, не ЗС, тк если от пользователя есть лайк и дизлайк одновременно - при объединении эта оценка распространится на каждое видео - получится сразу обе ценки для видео - не эквивалентное отношение
= R(IsLike, UserId) + R(IsLike, VideoId) - не ЗС
Как видно ни одно нетривиальное разбиение ЗС не является - отношение в 5-НФ.

Отношение <<Playlists created by User>>

Атрибуты:

- <<PlaylistId>> — идентификатор плейлиста
- <<UserId>> — идентификатор пользователя, который создал плейлист
- <<PlaylistName>> — название плейлиста

Функциональные зависимости:

- PlaylistId -> UserId
- Playlistid -> PlaylistName

Ключи:

- PlaylistId

Нормализация этого отношения:
все атрибуты атомарны и нету повторяющихся групп - отношение в 1-НФ
2ФЗ, зависят от ключа целиком - отношение в 2-НФ
ФЗ зависит от ключа напрямую - отношение в 3-НФ
ФЗ зависят от надключа - отношение в НФБК

4-НФ - для любой нетривиальной МЗ X->Y|Z, X - надключ. В отношении лишь 3 атрибута, честно перебрав все варианты X->Y|Z в МЗ можно убедиться, что все МЗ - тривиальны, значит отношение в 4НФ.
Перебор не привожу, тк там точно такие же рассуждения, как в отношении <<Video Likes>> - мы перебираем тройку и обнаруживаем, что множество Y не является независимым от Z. Не считаю нужным плодить копипасту.

5-НФ - для каждой нетривиальной ЗС каждый элемент декомпозиции - надключ. Честно переберем все возможные декомпозиции исходного отношения и проверим, что они могут быть ЗС. Перебор приводить не буду по тем же причинам. Среди всех вариантов разбиения лишь 1 ЗС является корректной - R(PlaylistId, UserId, PlaylistName) = R(PlaylistId, UserId) + R(PlaylistId, PlaylistName).
Заметим, что (PlaylistId, UserId) и (PlaylistId, PlaylistName) - надключи, а значит эта нетривиальная ЗС удовлетворяет условию - отношение в 5-НФ.

Денормализация:
Для удобства проверки условия в отношении <<Added videos>> - в качестве ключа я возьму надключ (PlaylistId, UserId). Объяснение ниже.

Отношение <<Added videos>>

Атрибуты:

- <<PlaylistId>> — идентификатор плейлиста
- <<UserId>> — идентификатор пользователя, который создал плейлист
- <<VideoId>> — идентификатор добавленного в плейлист видео

Функциональные зависимости:

- PlaylistId -> UserId

Ключи:

- (Playlistid, VideoId)

Нормализация этого отношения:
все атрибуты атомарны и нету повторяющихся групп - отношение в 1-НФ
В отношении PlaylistId -> UserId - UserId зависит не от ключа целиком. Потому декомпозируем
R(PlaylistId, UserId, VideoId) = Left(PlaylistId, UserId) + Right(PlaylistId, VideoId)
Теперь Left и Right во 2-НФ.
В обоих отношениях ФЗ зависит от ключа напрямую - отношения в 3-НФ
В обоих отношениях ФЗ зависят от надключа - отношения в НФБК

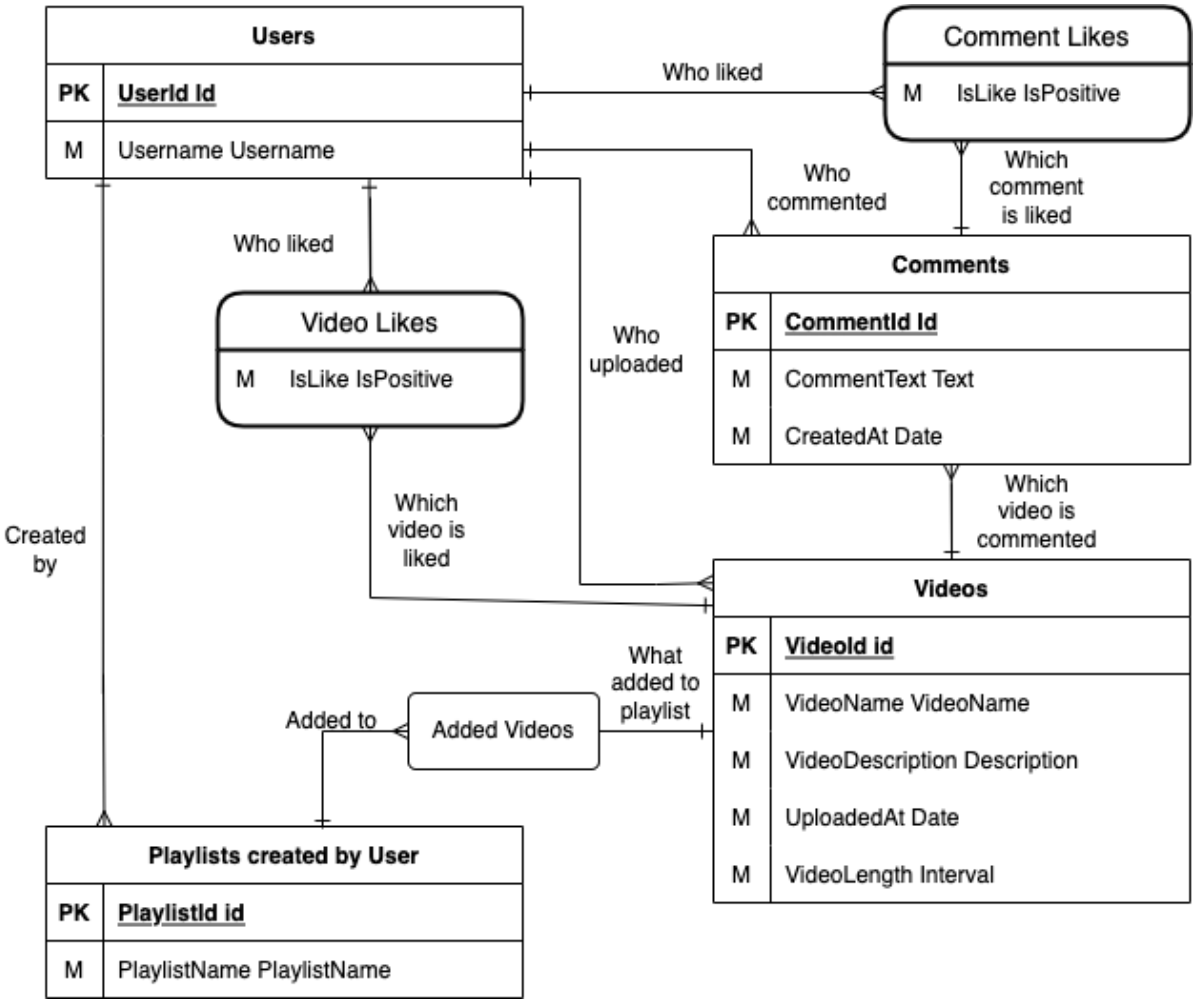
Отношение Left имеет единственный простой ключ - PlaylistId и находится в 3-НФ. По теореме Дейта-Фейгина 1 - оно находится в 5-НФ.

Отношение Right имеет два аттрибута - (PlaylistId, VideoId). В нем нету нетривиальных МЗ X->Y|Z хотя бы потому, что недостаточно аттрибутов для такого МЗ. Значит Right в 4-НФ.

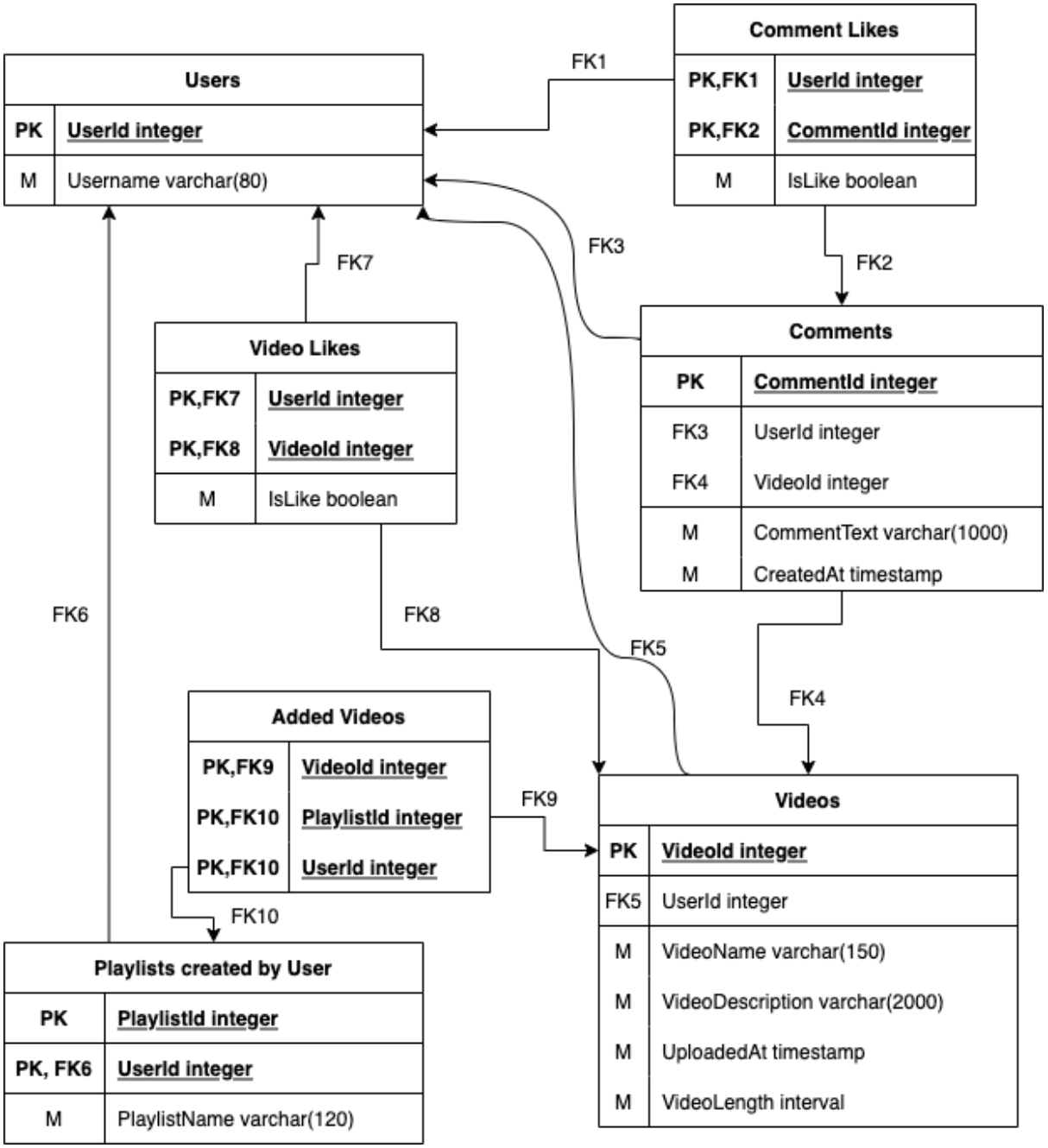
Также отношение Right не имеет ЗС, так как единственная нетривильная декмопозиция - R(PlaylistId, VideoId) => R(PlaylistId) + R(Videoid) не является корректной. Значит Right в 5-нф.

Денормализация:
Хоть отношения Left и Right находятся в 5-нф, они неудобны с точки зрения БД. На отношение <<Added videos>> хочется наложить ограничение, что в плейлист может добавлять только пользователь, который этот плейлист создал. Для этого в нем в качестве ключа можно использовать тройку (PlaylistId, UserId, VideoId), и (PlaylistId, UserId) будет внешним ключем на отношение <<Playlist created by User>>. Тогда для вставки VideoId в PlaylistId будет необходимо указывать вставляющего пользователя UserId, и БД будет сама проверять пару (PlaylistId, UserId) на корректность. Поэтому в качестве итогого отношения будет использоваться оригинальное <<Added videos>> с ключем (PlaylistId, UserId, Videoid).

Модель сущность-связь



Физическая модель



При построении физической модели использовалось следующее отображение доменов в типы:

Домен	Тип
`UserName`	`varchar(80)`
`Id`	`integer`
`IsPositive`	`boolean`
`Text`	`varchar(1000)`
`VideoName`	`varchar(150)`
`Description`	`varchar(2000)`
`Date`	`timestamp`
`Interval`	`interval`
`PlaylistName`	`varchar(120)`

Определения таблиц

Для реализации проекта использовалась СУБД psql (PostgreSQL) 14.1. Определения таблиц и их индексов приведено в файле `ddl.sql`.

Тестовые данные

Скрипт для добавления тестовых данных приведен в файле `data.sql`.

Запросы на получение данных

В рамках проекта были реализованы следующие запросы:

- Список всех видео
- Все видео, отсортированные по длительности
- Все видео пользователей, у которых в имени есть ‘itmo’
- Видео в плейлистах со словом ‘course’
- Количество лайков для каждого видео
- Количество комментариев для каждого видео
- Пары (комментарий, количество дизлайков) для всех комментариев
- Видео за год

Для реализации запросов были созданы вспомогательные представления:

- <<Hot Videos>> — эквивалент вкладки “В тренде”. Выдает видео, отсортированные по убыванию лайков, загруженные за последний день

Запросы на получение данных и вспомогательные представления приведены в файле `selects.sql`.

Запросы на изменение данных

В рамках проекта были реализованы следующие запросы:

- <<SetCommentLike>> — вспомогательная функция для следующих 2
- <<DislikeComment>> — ставит комментарию дизлайк от имени пользователя
- <<LikeComment>> — ставит комментарию лайк от имени пользователя

- <<SetVideoLike>> — вспомогательная функция для следующих 2
- <<DislikeVideo>> — ставит видео дизлайк от имени пользователя
- <<LikeVideo>> — ставит видео лайк от имени пользователя

- <<CreatePlaylist>> — создает плейлист с заданным именем у заданного пользователя
- <<CreateUser>> — создает пользователя с заданным именем именем
- <<UploadVideo>> — загружает видео от лица пользователя
- <<AddVideoToPlaylist>> — добавляет видео в плейлист пользователя, если он может в этот плейлист добавлять
- <<CommentVideo>> — оставить комментарий к видео от лица пользователя
- <<UpdatePlaylistName>> — обновить имя плейлиста от лица пользователя
- <<UpdateUserName>> — обновить имя пользователя
- <<UpdateVideoname>> — обновить название видео
- <<UpdateVideoDesription>> — обновить описание видео
- <<UpdateCommentText>> — обновить содержание комментария

Запросы на изменение данных, хранимые процедуры и триггеры приведены в файле `updates.sql`.